Collision Detection
2003-09-19

Datorgrafik

Stefan Seipel (stefan.seipel@hig.se)

Purposes of collision detection

Computer Animation and Offline-Rendering
- Modeling interaction of light, matter and objects
  (light tracing, shadow calculation)

Visual Simulation (engineering applications)
- Modeling interaction between objects in the scene
  (hit testing, event creation, physical behavior modeling)

Interactive Graphical Systems (VR, games)
- Modeling interaction between objects in the scene
  (hit testing, event creation, physical behavior modeling)
- Interacting between user and objects

Requirements to collision detection

Computer Animation and Offline-Rendering
- Highest possible accuracy
- Many parameters desired (hit, position, angles, texture coordinates)
- Timing aspect not so critical

Visual Simulation (engineering applications)
- Highest possible accuracy
- Many (physical) parameters desired (hit position, angles, velocity, mass-momentum)
- Timing aspects important

Interactive Graphical Systems (VR, games)
- Real-time aspect most important
- Accuracy often not predominant
- Little parameter density
- Physical behavior often approximated with simple functions

Collision Detection Approaches

Analytical Per-Primitive Methods:
- Every entity in a scene represented as parametric object (e.g. polygon, parametric function)
- Exact tests are performed based on linear algebra methods
- Accurate solutions, time consuming
- Final stage tests on pre-selected primitive subset

Approximation Methods:
- Discrete logic or algebraic solutions, low specificity, reliable for negative result
- Bounding volumes, simplified representations
- First stage testing to sort out non-intersecting objects
- Far-field testing

Hardware Supported Methods:
- Used hardware implemented features of transformation pipeline or rasterizer unit
- Require multi-pass rendering i.e. frame rate might drop
- May suffer from inaccuracy (whenever based on discrete buffers)

Analytical Hit Tests

- Point hits plane <=> Distance point - plane & sphere hits plane (static testing)
- Intersection ray and plane (vector based notion)
- Intersection ray and plane (parametric normal form)
- Sphere intersects with sphere
- Sphere hits plane (position prediction)
- Sphere hits sphere (position prediction)
- Intersection ray and triangle (based on vector based notion)
- Intersection ray and triangle (based on parametric solution)
- Triangle intersects Triangle

Point hits plane - distance point-plane - sphere hits plane

A) Assumption:
Plane with normal vector $N$ and point $P$ as the plane known.
Test point $T$ given.

Calculation of shortest distance between $T$ and plane:
There is a vector $PT$ from $P$ to $T$. The length of the projection of $PT$ upon $N$ can be calculated using the scalar product:

$$dT = (tx-x)*nx + (ty-y)*ny + (tz-z)*nz$$

B) Assumption:
Plane is given in normal form:
$Ax + By + Cz + D = 0$ where $A$, $B$, $C$ normal vector components, and shortest distance of plane to origin is known $D = -d$

Then insert $T$ in plane equation and check result: if $dT = 0$, then $T$ is on plane

$$d = Ax \times By - Cz + D = 0 ?$$
**Intersection ray and plane (vector based notion)**

Test with reliable result both for positive and negative result.

**Assumption:**
- Given three points \(v_1, v_2, \) and \(v_3\) on the plane.
- Given ray position \(P\) and direction vector \(d\).

**Span-vectors:**
- \(s_1 = v_2 - v_1\)
- \(s_2 = v_3 - v_1\)

Then:
\[
P + \tau \cdot d = v_1 + \lambda \cdot s_1 + \mu \cdot s_2
\]

Converse, Gauss

One solution \(\Rightarrow\) exactly one hit point

Many solutions \(\Rightarrow\) ray within plane

**Sphere intersect with sphere**

Collision test:
- Two spheres intersect if the distance of their midpoints is equal or less than the sum of their radius.

**Assumption:**
- \(C_1^*\) and \(C_2^*\) center coordinates of two spheres in \(R^3\) and radius \(R_1\) and \(R_2\) scalar values.

Then:
\[
d = |C_1 - C_2| \leq R_1 + R_2
\]

\[
\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2} \leq R_1 + R_2
\]

**Sphere hits plane (position prediction)**

**Known:**
- Sphere radius \(r\)
- Sphere center position \(CS\)
- Sphere motion direction \(D\)
- Point on plane and plane normal

**Approach:**
- If the sphere will hit the plane this will happen at point \(PS\) which is the point on the shortest way towards the plane.

Therefore:
\[
P = CS - r \cdot N
\]

\[
Hit = (CS - r \cdot N) + t \cdot D
\]

**Solution for \(t\):**
- Same as in ray-plane intersection testing (see above)

**Sphere hits sphere (position prediction)**

**Case of linear motion:**
- Theoretically the sweep volumes of the spheres intersect whenever shortest distance between their central lines is less than the sum of the radiuses.
- There is an analytical solution (see paper by Bard and Himel 2001).
- However, what about velocity aspects?
- Reverse calculation based on distance-to-hit, in order to calculate time window.
- What about non constant velocities?

**Case of non-linear motion track:**
- For a limited time interval into the future step through predictable positions of object 1.
- Use a time step resolution, which is adapted to the max. velocity under this time period.
- Otherwise possible hits might be left out.

**Time step resolution can be calculated so that e.g., the maximum allowed shift is less than 0.5 of the smallest radius of both objects.**
**Intersect ray and triangle (based on vector based notion)**

Test with reliable result both for positive and negative result.

Triangle : \( v_1, v_2, \) and \( v_3 \)

Span-vectors: 
\[ s_1 = v_2 - v_1 \]
\[ s_2 = v_3 - v_1 \]

Ray : \( P, \) and direction vector \( d \)

Then:
\[ P \cdot d + \lambda s_1 + \mu s_2 = \tau \]

\[ \lambda, \mu \in \mathbb{R} \] \( \lambda + \mu = 1 \)

Solution ?

Test with reliable result both for positive and negative result.

Method allows for texture coordinate generation!

---

**3D point inside triangle ?**

Parameter solution only yields a point on the plane of \( v_1 \ldots v_3 \).

Is the point inside the triangle ?

If the sum of the angles between
\[ a \]
\[ b \]
\[ c \]
\[ a + b + c = 360 \text{ degrees} \]
Then \( \text{hit is within the triangle} \)

\[ a \neq b \neq c \]

Where \( a \) tolerance value

Method allows NOT for texture coordinate generation!

---

**Triangle intersects triangle**

Test with reliable result both for positive and negative result.

Two triangles \( T_1, T_2 \) intersect, if at least one edge of \( T_1 \) intersects with \( T_2 \), or if one edge of \( T_2 \) intersects with \( T_1 \)

Perform 6 times a ray-triangle test and check LES result for:

\[ 0 \leq \lambda + \mu \leq 1 \]

Any edge of \( T_2 \) pierces \( T_1 \)? Any edge of \( T_1 \) pierces \( T_2 \)?

1. Test: \( T = v_1, v_2, v_3 \)
   \( R = v_4, (v_6-v_4) \)

2. Test: \( T = v_1, v_2, v_3 \)
   \( R = v_4, (v_5-v_4) \)

3. Test: \( T = v_1, v_2, v_3 \)
   \( R = v_5, (v_6-v_5) \)

4. Test: \( T = v_4, v_5, v_6 \)
   \( R = v_1, (v_3-v_1) \)

5. Test: \( T = v_4, v_5, v_6 \)
   \( R = v_1, (v_2-v_1) \)

6. Test: \( T = v_4, v_5, v_6 \)
   \( R = v_2, (v_3-v_2) \)

Test with reliable result both for positive and negative result.

Method allows for texture coordinate generation!

---

**Complexity considerations (e.g. triangle intersection testing)**

- Objects, and each object has \( n_{tri} \) triangles. Collision detection on a per-polygon basis.

Worst-Case scenario for triangle intersection test:

\[ \sum_{i=1}^{n_{poly}} n_{tri} \]

Example: Four objects containing 300, 400, 200, and 1000 triangles.
\[
\begin{align*}
N_1 & = 300*400 = 120000 \\
N_2 & = 400*300 = 120000 \\
N_3 & = 200*1000 = 200000 \\
N_4 & = 1000*1000 = 1000000
\end{align*}
\]

\[ \Rightarrow N = 1340000 \]

\[ \Rightarrow 6.6 \text{ Mio LES} \]

---

**Approximation approaches**

- Bounding box collision test
- Bounding sphere collision tests
- Hierarchical bounding volume tests (e.g. spheres)
- Multiple level-of-detail collision tests
- Spatial occupancy matrix and space voxelization
- Hot spot collision testing
- Collision test in image based rendering

---

**Bounding Box Intersection Testing : Initialization**

Bounding Box Calculation:

For any two objects \( A, B \) in the world coordinate system
- check relative transformation
- \( x_0 \) and \( y_0 \) translate
- \( z_0 \) rotate
- \( z_1 \) scale

\[ \begin{align*}
T_x & = v_1 + v_2 + v_3 \\
T_y & = v_4 + (v_6-v_4) \\
T_z & = v_4 + (v_5-v_4)
\end{align*} \]
Bounding Box Intersection Test

For any two bounding boxes A, B: Test all vertices in B for inside check A:

- \( (A_{mix} < B_{mix} < A_{max}) \)
- \( (A_{miy} < B_{miy} < A_{may}) \)
- \( (A_{miz} < B_{miz} < A_{maz}) \)

OR

- \( (A_{mix} < B_{mix} < A_{max}) \)
- \( (A_{miy} < B_{miy} < A_{may}) \)
- \( (A_{miz} < B_{maz} < A_{maz}) \)

OR

- \( (A_{mix} < B_{max} < A_{max}) \)
- \( (A_{miy} < B_{miy} < A_{may}) \)
- \( (A_{miz} < B_{maz} < A_{maz}) \)

OR

- \( (A_{mix} < B_{max} < A_{max}) \)
- \( (A_{miy} < B_{miy} < A_{may}) \)
- \( (A_{miz} < B_{maz} < A_{maz}) \)

Worst Case:

- 48 float point compares
- 23 logical operations

Bounding Box Intersection Testing : Comparison

Advantages:

- Simple arithmetic to test for collision
- Only bounding box corners need to be transformed on object motion
- Very fast test
- Bounding volume approximates elongated, flat objects quite well
- Negative test result is 100% reliable

Disadvantages:

- Bounding volume is not invariant with regard to rotation
- Object approximation can become quite in-accurate
- Positive test result is not specific

Bounding Sphere Intersection

Bounding Sphere Calculation:

1. For an object find the center of gravity \( C \) by calculating the average \( x, y, z \) components of all vertices.

\[
C = \left( \frac{\sum x_i}{n}, \frac{\sum y_i}{n}, \frac{\sum z_i}{n} \right)
\]

2. For all vertices in the object calculate the distance to \( C \) and keep the maximum distance value in mind. The maximum distance will become the bounding spheres radius.

Bounding Sphere Intersection Test (same as before)

Collision detection:

Test any two bounding spheres \( A \) and \( B \) of two objects for intersection. Two spheres intersect, if the distance of their midpoints is equal or less than the sum of their radii:

\[
d_{C_B - C_A} \leq R_A + R_B
\]

Hierarchical bounding volume tests

Approach:

- Use multiple (hierarchically) arranged bounding spheres to better approximate rigid objects
- Use multiple hierarchically arranged bounding sphere system to approximate articulated objects (e.g. robot arm systems...)

Tricky issue:

- Automatic generation of bounding sphere hierarchy
- Synchronization of model articulation and bounding sphere hierarchy update
Multiple level of detail (LOD) collision testing

Basic idea:
• Keep multiple geometric representations for a single object to be used in a scene
• Use models with high polygon count for close field-of-view rendering
• Use models with low polygon count for collision detection and far field of view rendering

Specific aspects:
• How much model degradation is allowable?
  • from a visual point of view
  • with regard to collision accuracy
• How much does it pay off?

Multiple LOD collision testing: Allowable degradation

<table>
<thead>
<tr>
<th>LOD</th>
<th>Triangles</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOD 1</td>
<td>7,446</td>
</tr>
<tr>
<td>LOD 2</td>
<td>3,724</td>
</tr>
<tr>
<td>LOD 3</td>
<td>746</td>
</tr>
<tr>
<td>LOD 4</td>
<td>500</td>
</tr>
</tbody>
</table>

Multiple LOD collision testing: Performance aspects

<table>
<thead>
<tr>
<th>Number of objects</th>
<th>Time to perform brute-force testing (0.1 microsecond/test)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>5,544</td>
</tr>
<tr>
<td>3</td>
<td>16,632</td>
</tr>
<tr>
<td>4</td>
<td>33,266</td>
</tr>
<tr>
<td>5</td>
<td>55,443</td>
</tr>
<tr>
<td>6</td>
<td>83,164</td>
</tr>
</tbody>
</table>

Spatial occupancy matrix and space voxelization

Example scenario: Many objects moving within a wide spatial region
(submarine scenario, terrain simulation, flight surveillance)

Goal: Avoiding the brute-force testing method (pairing all objects in scene)

Problem: To identify which object pairs are likely to collide?

Approach:
In order to make selection of object pairs which are likely to intersect, simulation space is partitioned in regularly arranged clusters.

At each simulation step, global positions for all objects are mapped upon cluster address.

Addressed cluster (e.g. voxel) keeps reference to objects within this cluster.

Cluster size and number depend on:
- maximum object size
- total extent of the scene
- memory available

Multiple LOD collision testing: Tricky aspects

Who performs mesh minimization?
Which mesh minimization criteria are applied?
Full enclosure envelope? False-Negative Results!
At what LOD does shape degrade too much?
Is model suitable for multiple LOD representation?

Multiple LOD collision testing: Hot-spot testing
Collision test in image based rendering

Objects are drawn as alpha blended textures
E.g. trees, bushes...
Analytical test yields positive result
However, hit is determined by texture fragment rendered at hit position
Calculation of texture coordinates required
Calculation of texel address
Lookup of texel value texture memory
Hit-coordinates in vector notation required

Hardware supported approaches

Pre-selection of objects in viewing volume
Picking using Selection Buffer (e.g. using OpenGL)
Picking using the BackBuffer (e.g. using OpenGL), Two Pass Rendering
Picking (analytical approach)

Pre-selection of visible objects using selection-buffer

Purpose:
- Reduce the number of tested objects (for analytical picking)
- Reduce the number of object-pairs (object collision testing)

Functional principle in OpenGL:
- Selection buffer, to carry names of primitives/objects drawn by OpenGL
- Name stack to push names
- OpenGL allows to load names onto the name stack
- Whenever under drawing a primitive passes the viewing frustum test the current name on the name stack is added to the selection buffer
  (if that is different from the last name in the selection buffer)

Pre-selection of visible objects using selection-buffer

Selection record structure

- Number n of names on stack, place for names
- Name stack pointers (structure of names)

Our example:
- Name stack
- Selection buffer

Picking objects using the selection-buffer

Purpose: Identifying object beyond current cursor position

Special case of selection:
For a given cursor position on screen, a narrow viewing volume can be set up.
Picking objects using the back-buffer

Purpose: Identifying object beyond current cursor position

- Render full featured objects to back-buffer (i.e. lightning, material etc. activated)
- Swap buffers (visual result evident in front-buffer)
- Disable shading, illumination
- Encode object(s) identifier into color code and set rawing color
- Render object(s) into back buffer
- Read pixels from back buffer at cursor position (glReadPixles)
- Decode color code into object identifier

Features:
- Two-pass rendering approach, reduces effective frame rate
- Sometimes complicated implementation approach
- Utilized HW supported rasterizing and geometry transformation
- Excellent hardware FoV culling

Hardware Approaches: Discussion

- Very special pointing metaphor
- Ray: eye-cursor-target
- Rotational ray-control in screen space
- Hit-tests in screen domain
- No hits outside viewing volume
- No hits for culled polygons
- Easy to control with a 2D mouse
- Unusable for full scale or immersive VR

Alternative Solution:
- Picking ray modeled as part of the 3D scene
- Collision testing using analytical ray-object testing

Discrete techniques using regular grids

Event maps / semantic maps
- Digital elevation maps
- Distance maps

Semantic maps / event maps

Efficient Collision Detection / Model Representation

Digital elevation maps (DEMs)

Example application: Terrain simulation scenarios

Assumption: Terrain is primarily given as a regular two-dimensional grid of height values E[n,m]

Simulation consequences:
- DEM cannot be rendered immediately within the 3D simulation. Therefore, a polygonal mesh must be re-created from the 2.5-dimensional elevation map.
- Two types of polygonal meshes.

Approach:
- Find a suitable transfer function, which maps 3D simulation coordinates into 2.5D DEM coordinates.
- For a point in discrete row and column index pair (i,k), a coordinate in simulation coordinates space, there is plane parallel to the xz-plane.

Collision test:
1. Look-up elevation value at e = E[i,k] and compare y for interval [e-ε...e+ε], where ε is a predefined fuzz value.
2. Look-up polygons in the terrain model which correspond to this matrix column/row and perform additional tests.

Digital elevation maps (DEMs) continued

The role of e:
- Mapping e.g.:
  \( e = E[n,m] \)
- Mapping from real world coordinates to matrix indexing causes aliasing
  \( \text{Value } E[n,m] \text{ has indeterminate elevation in the neighborhood of }\)
  \( \text{the actual point } p \)
  \( \text{if } \text{dem scans vertical tolerance area to compensate for spatial aliasing} \)
  \( \text{depending on largest possible quantisation error in } x, y, \text{plane i.e. 0.5 grid cell size} \)
  \( \text{depending on largest slope across cell} \)
  \( \text{controls the sensitivity of the hit test} \)
Distance maps

Application example: Occlusal simulation

Problem:
• Extremely high polygon count requires other strategies than polygon intersection testing.

Approach:
• Have or create 2.5 dimensional representation for each object
• Distance map with high resolution where elevation coincides with the main direction of motion
• Implement a rasterizer engine which transforms one elevation map into the other one
• Per-pixel comparison of elevation values and collision detection

Performance:
Mapsize = 900x600 = 0.54 Mio. voxels
Time for transformation and test: 0.745 sec. (300Mhz PIII)
Comparison:
Analytical polygon intersection testing (1.1 millisecond per test)
Approx. 2 hours 30 minutes.

Risk:
Discrete technique is subject to aliasing artifacts
Can be reduced using high-resolution distance maps.
Collision Testing Strategies: Object-Object

1. Step: Static collision management
   Q: Which object are of interest for collision detection?
   A: Usually predefined in system build phase

2. Step: Dynamic collision management
   Q: Which object pairs must be considered for collision detection at current simulation time?
   M: Spatial occupancy matrix, selection techniques, dynamics prediction
   A: List of paired objects of interest which might collide

3. Step: Sorting out the true-negative pairs
   Q: Which pairs of objects do certainly not intersect?
   M: Bounding volume testing, reduced LOD testing in some cases
   A: Reduced list of paired objects of interest which might collide

4. Step: Find true-positive pairs
   Q: Which objects do actually intersect?
   At which spatial location do they intersect?
   Texture coordinates?
   Impact angle?
   M: The entire palette of analytical tests